درس طراحی زبان‌های برنامه‌سازی

دکتر محمد ایزدی

تمرین چهارم

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال دوم ۱۴۰۰ـ۱۳۹۹

مهلت ارسال:
۲۸ اردیبهشت ۱۴۰۰
ساعت ۲۳:۵۹

به موارد زیر توجه کنید:

* برنامه‌های خود را به زبان Racket بنویسید.
* مهلت ارسال تمرین ساعت ۲۳:۵۹ روز ۲۸ اردیبهشت ۱۴۰۰ است.
* جواب خود را در غالب pdf بنویسید. در صورت نیاز می‌توانید کدهای خود را در کنار pdf بگذارید.
* در نهایت تمام فایل‌های خود را در یک فایل زیپ قرار داده و با نام $HW4\_StudentID$ در سامانه کوئرا آپلود کنید.
* هرگونه سوالی راجع به این تمرین را در زیر پست مربوطه در کوئرای درس مطرح کنید.
* در مجموعِ تمامی تمارین ۷ روز مهلت تاخیر مجاز دارید و پس از تمام شدن این تاخیرهای مجاز به ازای هر روز ۱۰ درصد از کل نمره تمرین شما کم می‌شود.
* لطفا تمرین‌ها را از یکدیگر کپی نکنید. در صورت وقوع چنین مواردی مطابق با سیاست درس رفتار می‌شود.
* بخش سوال‌های پیشنهادی نیازی به تحویل ندارد و صرفا جهت تمرین بیشتر گذاشته شده است.

نکته مهم : در سوال‌هایی که از شما خواسته می‌شود امکان جدیدی را به یک زبان اضافه کنید، باید تمام تغییرات روی گرامر، توصیف مقادیر [۱] ، محیط [۲] ، قوانین استنتاج [۳] دستورات و کدهای تغییر یافته یا اضافه شده را مشخص نمایید. (بعضی از این موارد ممکن است تغییر نکنند.) در واقع باید همه مراحل پیاده‌سازی را بنویسید. دقت کنید تغییرات روی سینتکس نیازی به بررسی ندارند.

سوال ۱

برای شکل 3.5 صفحه‌ی 68 کتاب یک درخت استخراج [۴] بکشید.

سوال ۲

به زبان let معرفی شده در کلاس، تابع جدید توان را اضافه کنید. این تابع به شکل $pow(exp1, exp2)$ است و عدد اول را به توان عدد دوم می‌رساند. (هر دو عدد حقیقی‌اند.)

سوال ۳

نوع داده‌ی لیست را به این زبان اضافه کنید. توابع قبلی (مانند جمع و ...) روی این نوع داده عمل نمی‌کنند. توابع جدید زیر را برای این نوع داده پیاده‌سازی کنید.

- empty-list − return an empty list
- car − like car in Racket
- cdr − like cdr in Racket
- null? − like null? in Racket
- concatenate − concatenate two lists and return the result
- sort − return a list that is sorted of the input list.

نحوه‌ی پیاده‌سازی این توابع بر عهده‌ی خودتان است.

[1] Specification of Values
[2] Environment
[3] Rules of inference
[4] derivation tree

سوال ۴

به زبان ،let تابع letc را اضافه کنید.

$$letc\ \{Identifier = Expression\}^*\ in\ Expression$$

این تابع دقیقا همانند let است ولی به جای یک ورودی چند ورودی می‌گیرد. این تابع همانند چند let پشت سر هم عمل می‌کند بنابراین تغییرات اولین تساوی روی ادامه‌ی تساوی‌ها تاثیر می‌گذارد. به طور مثال:

ورودی نمونه

```
letc x = 2 x = x - 1 y = x in y
```

خروجی نمونه

```
1
```

سوال ۵

با استفاده از زبان proc و بدون استفاده از تابع letrec برنامه‌ای بنویسید که عدد n ام دنباله‌ی فیبوناچی را تولید کند. در واقع این برنامه یک proc دارد که ورودی n را گرفته و عدد n ام دنباله‌ی فیبوناچی را تولید می‌کند.

سوال ۶

به زبان letrec تابع letrecc را اضافه کنید. این تابع دقیقا همانند letrec است با این تفاوت که به جای یک تابع، چند تابع می‌توانند در آن تعریف شوند. دقت کنید تمام این توابع باید همدیگر را بشناسند و بتوانند از هم استفاده کنند.

سوال‌های پیشنهادی

سوال‌های زیر از کتاب $essentials\ of\ programming\ languages$ توصیه می‌شود:

3.8, 3.13, 3.18, 3.21, 3.26